Smooth adversarial examples

Hanwei Zhang, Yannis Avrithis, Teddy Furon, Laurent Amsaleg

▶ **To cite this version:**

Hanwei Zhang, Yannis Avrithis, Teddy Furon, Laurent Amsaleg. Smooth adversarial examples. EURASIP Journal on Information Security, 2020, 2020 (1), 10.1186/s13635-020-00112-z . hal-03017171

HAL Id: hal-03017171
https://hal.science/hal-03017171

Submitted on 8 Dec 2020

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Smooth adversarial examples

Hanwei Zhang[1,2*], Yannis Avrithis[1], Teddy Furon[1] and Laurent Amsaleg[1]

Abstract

This paper investigates the visual quality of the adversarial examples. Recent papers propose to smooth the perturbations to get rid of high frequency artifacts. In this work, smoothing has a different meaning as it perceptually shapes the perturbation according to the visual content of the image to be attacked. The perturbation becomes locally smooth on the flat areas of the input image, but it may be noisy on its textured areas and sharp across its edges. This operation relies on Laplacian smoothing, well-known in graph signal processing, which we integrate in the attack pipeline. We benchmark several attacks with and without smoothing under a white box scenario and evaluate their transferability. Despite the additional constraint of smoothness, our attack has the same probability of success at lower distortion.

Keywords: Adversarial example, Image classification, Deep neural network

1 Introduction

Adversarial examples where introduced by Szegedy et al. [1] as *imperceptible* perturbations of a test image that can change a neural network's prediction. This has spawned active research on adversarial attacks and defenses with competitions among research teams [2]. Despite the theoretical and practical progress in understanding, the sensitivity of neural networks to their input, assessing the imperceptibility of adversarial attacks remains elusive: user studies show that L_p norms are largely unsuitable, whereas more sophisticated measures are limited too [3].

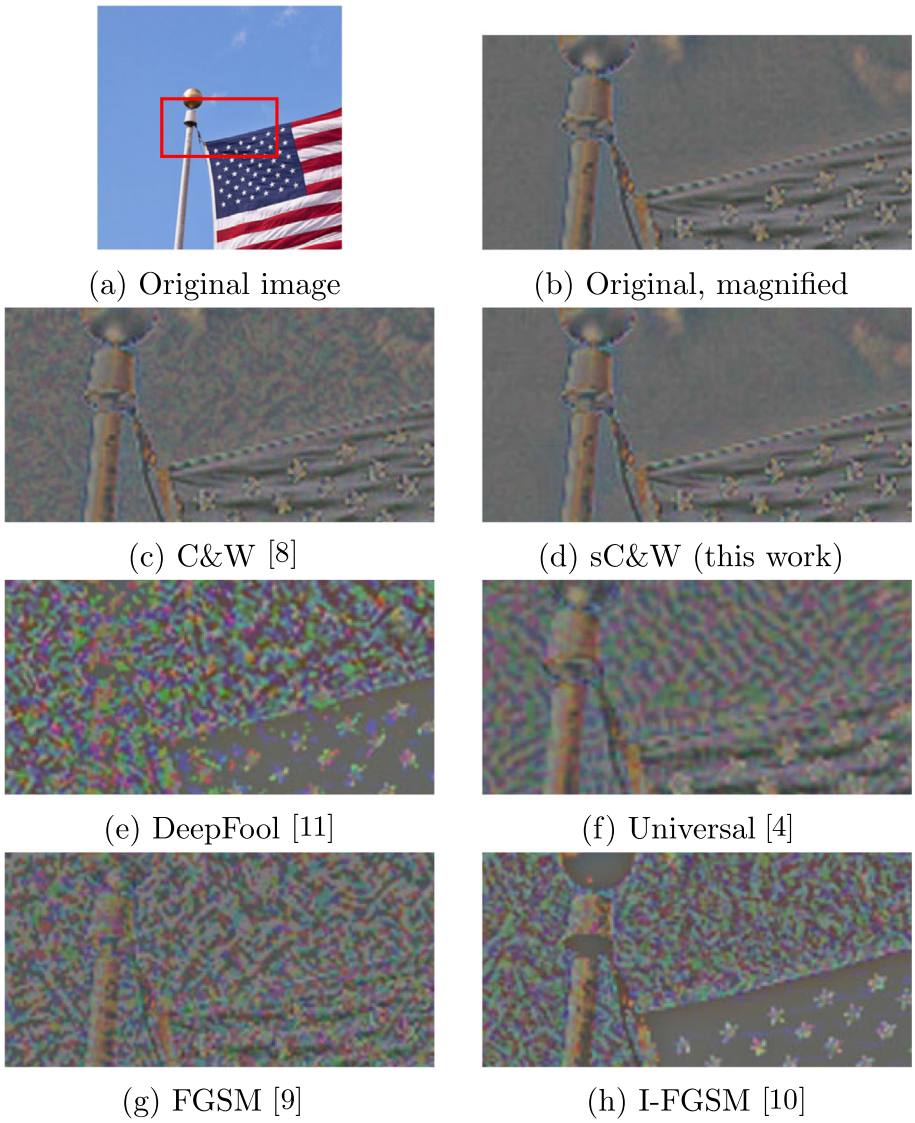
Machine assessment of perceptual similarity between two images (the input image and its adversarial example) is arguably as difficult as the original classification task, while human assessment of whether one image is adversarial is hard when the L_p norm of the perturbation is small. Of course, when both images are available and the perturbation is isolated, one can always see it. To make the problem interesting, we ask the following question: *given a single image, can the effect of a perturbation be magnified to the extent that it becomes visible and a human may decide whether this example is benign or adversarial?*

Figure 1 shows that the answer is positive for a range of popular adversarial attacks. In Appendix 1, we propose a simple *adversarial magnification* producing a "magnified" version of a given image, without the knowledge of any other reference image. Assuming that natural images are locally smooth, this can reveal not only the existence of an adversarial perturbation but also its pattern. One can recognize, for instance, the pattern of Fig. 4 of [4] in our Fig. 1f, revealing a universal adversarial perturbation.

Motivated by this example, we argue that popular adversarial attacks have a fundamental limitation in terms of imperceptibility that we attempt to overcome by introducing *smooth adversarial examples*. Our attack assumes local smoothness and generates examples that are consistent with the precise smoothness pattern of the input image. More than just looking "natural" [5] or being smooth [6, 7], our adversarial examples are photorealistic, low-distortion, and virtually invisible even under magnification. This is evident by comparing our magnified example in Fig. 1d to the magnified original in Fig. 1b.

Given that our adversarial examples are more constrained, an interesting question is whether they perform well according to metrics like probability of success and L_p distortion. We show that our attack not only is

*Correspondence: hanwei.zhang@irisa.fr
[1]Univ Rennes, Inria, CNRS, IRISA, Campus de Beaulieu, Rennes, France
[2]East China Normal University, North Zhongshan Road Campus, Shanghai, China



(a) Original image (b) Original, magnified

(c) C&W [8] (d) sC&W (this work)

(e) DeepFool [11] (f) Universal [4]

(g) FGSM [9] (h) I-FGSM [10]

Fig. 1 Given a single input image, our *adversarial magnification* (*cf* Appendix 1) reveals the effect of a potential adversarial perturbation. We show **a** the original image followed by **b** its own magnified version as well as **c**–**h** magnified versions of adversarial examples generated by different attacks. Our *smooth adversarial example* (**d**) is invisible even when magnified

competitive but outperforms Carlini & Wagner [8], from which our own attack differs basically by a smoothness penalty.

1.1 Contributions

As *primary* contributions, we

1 Investigate the behavior of existing attacks when perturbations become "smooth like" the input image; and
2 Devise one attack that performs well on standard metrics while satisfying the new constraint.

As *secondary* contributions, we

3. *Magnify* perturbations to facilitate qualitative evaluation of their imperceptibility;
4. Show that properly integrating the smoothness constraint is not as easy as smoothing the perturbation generated by an attack; and
5. Define a new, more complete/fair *evaluation protocol*.

The remaining text is organized as follows. Section 2 formulates the problem and introduces a classification of attacks. It describes the C&W attack and the related work. Section 3 explains Laplacian smoothing, on which we build our method. Section 4 presents our *smooth*

adversarial attacks, and Section 5 provides experimental evaluation. Conclusions are drawn in Section 6. Our *adversarial magnification* used to generate Fig. 1 is specified in Appendix 1.

2 Problem formulation and related work

Let us denote by $\mathbf{x} \in \mathcal{X} := [0,1]^{n \times d}$ an image of n pixels and d color channels that has been flattened in a given ordering of the spatial components. A classifier network f maps that input image $\mathbf{x}$ to an output $\mathbf{y} = \mathsf{f}(\mathbf{x}) \in \mathbb{R}^k$ which contains the *logits* of k classes. It is typically followed by softmax and cross-entropy loss at supervised training or by arg max at test time. An input $\mathbf{x}$ with logits $\mathbf{y} = \mathsf{f}(\mathbf{x})$ is correctly classified if the *prediction* $\mathsf{p}(\mathbf{x}) := \arg\max_i y_i$ equals the true label of $\mathbf{x}$.

The attacker mounts a *white box* attack that is specific to f, public and known. The attack modifies an original image $\mathbf{x}_o \in \mathcal{X}$ with given true label $t \in \{1, \dots, k\}$ into an adversarial example $\mathbf{x}_a \in \mathcal{X}$, which may be incorrectly classified by the network, that is $\mathsf{p}(\mathbf{x}_a) \neq t$, although it looks similar to the original $\mathbf{x}_o$. The latter is often expressed by a small L_2 *distortion* $\|\mathbf{x}_a - \mathbf{x}_o\|$.

2.1 Families of attacks

In a white box setting, attacks typically rely on exploiting the gradient of some loss function. We propose to classify known attacks into three families.

2.1.1 Target distortion

This family gathers attacks targeting a distortion ϵ given as an input parameter. Examples are early attacks like Fast Gradient Sign Method (FGSM) [9] and Iterative-FGSM (I-FGSM) [10]. Their performance is then measured by the *probability of success* $P_{\text{suc}} := \mathbb{P}(\mathsf{p}(\mathbf{x}_a) \neq t)$ as a function of ϵ.

2.1.2 Target success

This family gathers attacks that always succeed in misclassifying $\mathbf{x}_a$, at the price of a possible large distortion. DeepFool [11] is a typical example. Their performance is then measured by the *expected distortion* $\overline{D} := \mathbb{E}(\|\mathbf{x}_a - \mathbf{x}_o\|)$.

These two first families are implemented with variations of a gradient descent method. A *classification loss* function is defined on an output logit vector $\mathbf{y} = \mathsf{f}(\mathbf{x})$ with respect to the original true label t, denoted by $\ell(\mathbf{y}, t)$.

2.1.3 Target optimality

The above attacks are not optimal because they a priori do not solve the problem of succeeding under minimal distortion,

$$\min_{\mathbf{x} \in \mathcal{X} : \mathsf{p}(\mathbf{x}) \neq t} \|\mathbf{x} - \mathbf{x}_o\|. \tag{1}$$

Szegedy et al. [1] approximate this constrained minimization problem by a Lagrangian formulation

$$\min_{\mathbf{x} \in \mathcal{X}} \lambda \, \|\mathbf{x} - \mathbf{x}_o\|^2 + \ell(\mathsf{f}(\mathbf{x}), t). \tag{2}$$

Parameter λ controls the trade-off between the distortion and the classification loss. Szegedy et al. [1] carry out this optimization by box-constrained L-BFGS.

The attack of Carlini & Wagner [8], denoted C&W in the sequel, pertains to this approach. A change of variable eliminates the box constraint: $\mathbf{x} \in \mathcal{X}$ is replaced by $\sigma(\mathbf{w})$, where $\mathbf{w} \in \mathbb{R}^{n \times d}$ is a latent vector and σ is the element-wise sigmoid function that projects $\mathbb{R}^{n \times d}$ to $\mathcal{X}$. A margin is introduced: an *untargeted attack* makes the logit y_t less than any other logit y_i for $i \neq t$ by at least a margin $m \geq 0$. Similar to the multi-class SVM loss by Crammer and Singer [12] (where $m = 1$), the loss function ℓ is then defined as

$$\ell(\mathbf{y}, t) := [\, y_t - \max_{i \neq t} y_i + m]_+ , \tag{3}$$

where $[\cdot]_+$ denotes the positive part. The C&W attack uses the Adam optimizer [13] to minimize the functional

$$J(\mathbf{w}, \lambda) := \lambda \, \|\sigma(\mathbf{w}) - \mathbf{x}_o\|^2 + \ell(\mathsf{f}(\sigma(\mathbf{w})), t), \tag{4}$$

initializing by $\mathbf{w}_o := \sigma^{-1}(\mathbf{x}_o)$. When the margin is reached, loss $\ell(\mathbf{y}, t)$ (3) vanishes and the distortion term pulls $\sigma(\mathbf{w})$ back towards $\mathbf{x}_o$, causing oscillations around the margin. Among all successful iterates, the one with the least distortion is kept; if there is none, the attack fails. The process is repeated for different Lagrangian multiplier λ according to line search. This family of attacks is typically more expensive than the two first.

2.2 Imperceptibility of adversarial perturbations

Adversarial perturbations are often invisible only because their amplitude is extremely small. Few papers deal with the need of improving the imperceptibility of the adversarial perturbations. The main idea in this direction is to create low or mid-frequency perturbation patterns.

Zhou et al. [14] add a regularization term for the sake of transferability, which removes the high frequencies of the perturbation via low-pass spatial filtering. Heng et al. [6] propose a harmonic adversarial attack where perturbations are very smooth gradient-like images. Guo et al. [7] design an attack explicitly in the Fourier domain. However, in all cases above, the convolution and the bases of the harmonic functions and of the Fourier transform are independent of the visual content of the input image.

In contrast, the adversarial examples in this work are crafted to be locally compliant with the smoothness of the original image. Our perturbation may be sharp across the edges of $\mathbf{x}_o$ but smooth wherever $\mathbf{x}_o$ is, e.g., on background regions. It is not just smooth but photorealistic,

because its smoothness pattern is guided by the input image.

An analogy becomes evident with *digital watermarking* [15]. In this application, the watermark signal pushes the input image into the detection region (the set of images deemed as watermarked by the detector), whereas here the adversarial perturbation drives the image outside its class region. The watermark is invisible thanks to the masking property of the input image [16]. Its textured areas and its contours can hide a lot of watermarking power, but the flat areas cannot be modified without producing noticeable artifacts. Perceptually shaping the watermark signal allows a stronger power, which in turn yields more robustness.

Another related problem, with similar solutions mathematically, is photorealistic *style transfer*. Luan et al. [17] transfer style from a reference style image to an input image, while constraining the output to being photorealistic with respect to the input. This work as well as follow-up works [18, 19] is based on variants of Laplacian smoothing or regularization much like we do.

It is important to highlight that high frequencies can be powerful for deluding a network, as illustrated by the extreme example of the *one pixel attack* [20]. However, this is arguably one of the most visible attacks.

3 Background on graph Laplacian smoothing

Popular attacks typically produce noisy patterns that are not found in natural images. They may not be visible at first sight because of their low amplitude, but they are easily detected once magnified (see Fig. 1). Our objective is to craft an adversarial perturbation that is locally as smooth as the input image, remaining invisible through magnification. This section gives background on *Laplacian smoothing* [21, 22], a classical operator in *graph signal processing* [23, 24], which we adapt to images here. Section 4 uses it to generate a smooth perturbation guided by the original input image.

3.1 Graph

Laplacian smoothing builds on a weighted undirected graph whose n vertices correspond to the n pixels of the input image $\mathbf{x}_o$. The ith vertex of the graph is associated with feature $\mathbf{x}_i \in [0,1]^d$ that is the ith row of $\mathbf{x}_o$, that is, $\mathbf{x}_o = [\,\mathbf{x}_1, \ldots, \mathbf{x}_n]^\top$. Matrix $\mathbf{p} \in \mathbb{R}^{n\times 2}$ denotes the spatial coordinates of the n pixels in the image, and similarly $\mathbf{p} = [\,\mathbf{p}_1, \ldots, \mathbf{p}_n]^\top$. An edge (i,j) of the graph is associated with weight $w_{ij} \geq 0$, giving rise to an $n \times n$ symmetric adjacency matrix $\mathbf{W}$, for instance defined as

$$w_{ij} := \begin{cases} \mathsf{k}_\mathsf{f}(\mathbf{x}_i, \mathbf{x}_j)\mathsf{k}_\mathsf{s}(\mathbf{p}_i, \mathbf{p}_j), & \text{if } i \neq j \\ 0, & \text{if } i = j \end{cases} \tag{5}$$

for $i,j \in \{1, \ldots, n\}$, where k_f is a *feature kernel* and k_s is a *spatial kernel*, both being usually Gaussian or Laplacian. The spatial kernel is typically nonzero only on nearest neighbors, resulting in a sparse matrix $\mathbf{W}$. We further define the $n \times n$ *degree matrix* $\mathbf{D} := \mathrm{diag}(\mathbf{W}\mathbf{1}_n)$ where $\mathbf{1}_n$ is the all-ones n-vector.

3.2 Regularization [21]

Now, given a new signal $\mathbf{z} \in \mathbb{R}^{n\times d}$ on this graph, the objective of graph smoothing is to find another signal $\mathbf{r}$, which is close to $\mathbf{z}$, while at the same time being smooth according to the neighborhood system represented by the graph. Precisely, given $\mathbf{z}$, we define the *output signal* $\mathsf{s}_\alpha(\mathbf{z}) := \arg\min_{\mathbf{r}\in\mathbb{R}^{n\times d}} \phi_\alpha(\mathbf{r}, \mathbf{z})$, with

$$\phi_\alpha(\mathbf{r}, \mathbf{z}) := \frac{\alpha}{2} \sum_{i,j} w_{ij} \left\| \hat{\mathbf{r}}_i - \hat{\mathbf{r}}_j \right\|^2 + (1-\alpha) \left\| \mathbf{r} - \mathbf{z} \right\|_F^2 \tag{6}$$

where $\hat{\mathbf{r}} := \mathbf{D}^{-1/2}\mathbf{r}$ and $\|\cdot\|_F$ is the Frobenius norm. The first summand is the *smoothness term*. It encourages $\hat{\mathbf{r}}_i$ to be close to $\hat{\mathbf{r}}_j$ when w_{ij} is large, i.e., when pixels i and j of input $\mathbf{x}_o$ are neighbors and similar. This encourages $\mathbf{r}$ to be smooth wherever $\mathbf{x}_o$ is. The second summand is the *fitness term* that encourages $\mathbf{r}$ to stay close to $\mathbf{z}$. Parameter $\alpha \in [\,0, 1)$ controls the trade-off between the two.

3.3 Filtering

If we symmetrically normalize matrix $\mathbf{W}$ as $\mathcal{W} := \mathbf{D}^{-1/2}\mathbf{W}\mathbf{D}^{-1/2}$ and define the $n \times n$ *regularized Laplacian* matrix $\mathcal{L}_\alpha := (\mathbf{I}_n - \alpha\mathcal{W})/(1-\alpha)$, then the expression (6) simplifies to the following quadratic form:

$$\phi_\alpha(\mathbf{r}, \mathbf{z}) = (1-\alpha)\mathrm{tr}\left(\mathbf{r}^\top \mathcal{L}_\alpha \mathbf{r} - 2\mathbf{z}^\top \mathbf{r} + \mathbf{z}^\top \mathbf{z}\right). \tag{7}$$

This reveals, by letting the derivative $\partial\phi/\partial\mathbf{r}$ vanish independently per column, that the smoothed signal is given in closed form:

$$\mathsf{s}_\alpha(\mathbf{z}) = \mathcal{L}_\alpha^{-1}\mathbf{z}. \tag{8}$$

This solution is unique because matrix $\mathcal{L}_\alpha$ is positive-definite. Parameter α controls the bandwidth of the smoothing: function s_α is the all-pass filter for $\alpha = 0$ and becomes a strict "low-pass" filter when $\alpha \to 1$ [25].

Variants of the model above have been used for instance for interactive image segmentation [22, 26, 27], transductive semi-supervised classification [21, 28], and ranking on manifolds [29, 30]. Input $\mathbf{z}$ expresses labels known for some input pixels (for segmentation) or samples (for classification), or identifies queries (for ranking), and is null for the remaining vertices. Smoothing then spreads the labels to these vertices according the weights of the graph.

3.4 Normalization

Contrary to applications like interactive segmentation or semi-supervised classification [21, 22], $\mathbf{z}$ does not represent a binary labeling but rather an arbitrary perturbation

in this work. Also contrary to such applications, the output is neither normalized nor taken as the maximum over feature dimensions (channels). If $\mathcal{L}_\alpha^{-1}$ is seen as a spatial filter, we therefore row-wise normalize it to one in order to preserve the dynamic range of $\mathbf{z}$. We therefore define the *normalized smoothing function* as

$$\hat{\mathsf{s}}_\alpha(\mathbf{z}) := \operatorname{diag}(\mathsf{s}_\alpha(\mathbf{1}_n))^{-1}\mathsf{s}_\alpha(\mathbf{z}). \tag{9}$$

This function of course depends on $\mathbf{x}_o$. We omit this from notation but we say $\hat{\mathsf{s}}_\alpha$ is *smoothing guided* by $\mathbf{x}_o$ and the output is *smooth like* $\mathbf{x}_o$.

4 Integrating smoothness into the attack

The key idea of the paper is that the smoothness of the perturbation is now consistent with the smoothness of the original input image $\mathbf{x}_o$, which is achieved by smoothing operations guided by $\mathbf{x}_o$. This section integrates smoothness into attacks targeting distortion (Section 4.1) and attacks targeting optimality (Section 4.2), but in very different ways.

4.1 Simple attacks

We consider here simple attacks targeting distortion or success based on gradient descent of the loss function. There are many variations which normalize or clip the update according to the norm used for measuring the distortion, a learning rate or a fixed step etc. These variants are loosely prototyped as the iterative process

$$\mathbf{g} = \nabla_{\mathbf{x}}\ell\left(\mathsf{f}(\mathbf{x}_a{}^{(k)}), t\right), \tag{10}$$

$$\mathbf{x}_a{}^{(k+1)} = \mathsf{c}\left(\mathbf{x}_a{}^{(k)} - \mathsf{n}(\mathbf{g})\right), \tag{11}$$

where c is a *clipping* function and n a *normalization* function according to the variant. Function c should at least produce a valid image: $\mathsf{c}(\mathbf{x}) \in \mathcal{X} = [0,1]^{n\times d}$.

4.1.1 Quick and dirty

To keep these simple attacks simple, smoothness is loosely integrated after the gradient computation and before the update normalization:

$$\mathbf{x}_a{}^{(k+1)} = \mathsf{c}\left(\mathbf{x}_a{}^{(k)} - \mathsf{n}(\hat{\mathsf{s}}_\alpha(\mathbf{g}))\right). \tag{12}$$

This approach can be seen as a projected gradient descent on the manifold of perturbations that are smooth like $\mathbf{x}_o$. When applied to PGD_2, we call this attack qPGD_2 where the "q" stands for a "quick and dirty" integration of the smoothness constraint.

4.2 Attack targeting optimality

This section integrates smoothness in the attacks targeting optimality like C&W. Our starting point is the unconstrained problem (4) [8]. However, instead of representing the perturbation signal $\mathbf{r} := \mathbf{x} - \mathbf{x}_o$ implicitly as a function $\sigma(\mathbf{w}) - \mathbf{x}_o$ of another parameter $\mathbf{w}$, we express the objective explicitly as a function of variable $\mathbf{r}$, as in the original formulation of (2) in [1]. We make this choice because we need to directly process the perturbation $\mathbf{r}$. On the other hand, we now need the element-wise clipping function $\mathsf{c}(\mathbf{x}) := \min([\mathbf{x}]_+, 1)$ to satisfy the constraint $\mathbf{x} = \mathbf{x}_o + \mathbf{r} \in \mathcal{X}$ (2). Our problem is then

$$\min_{\mathbf{r}} \quad \lambda\|\mathbf{r}\|^2 + \ell(\mathsf{f}(\mathsf{c}(\mathbf{x}_o + \mathbf{r})), t), \tag{13}$$

where $\mathbf{r}$ is unconstrained in $\mathbb{R}^{n\times d}$.

4.2.1 Smoothness penalty

At this point, optimizing (13) results in "independent" updates at each pixel. We would rather like to take the smoothness structure of the input $\mathbf{x}_o$ into account and impose a similar structure on $\mathbf{r}$. Representing the pairwise relations by a graph as discussed in Section 3, a straightforward choice is to introduce a pairwise loss term

$$\mu \sum_{i,j} w_{ij} \left\|\hat{\mathbf{r}}_i - \hat{\mathbf{r}}_j\right\|^2 \tag{14}$$

into (13), where we recall that w_{ij} are the elements of the adjacency matrix $\mathbf{W}$ of $\mathbf{x}_o$, $\hat{\mathbf{r}} := \mathbf{D}^{-1/2}\mathbf{r}$ and $\mathbf{D} := \operatorname{diag}(\mathbf{W}\mathbf{1}_n)$. A problem is that the spatial kernel is typically narrow to capture smoothness only locally. Even if parameter μ is large, it would take a lot of iterations for the information to propagate globally, each iteration needing a forward and backward pass through the network.

4.2.2 Smoothness constraint

What we advocate instead is to apply a global smoothing process at each iteration: we introduce a *latent variable* $\mathbf{z} \in \mathbb{R}^{n\times d}$ and seek for a joint solution with respect to $\mathbf{r}$ and $\mathbf{z}$ of the following

$$\min_{\mathbf{r},\mathbf{z}} \quad \mu\phi_\alpha(\mathbf{r},\mathbf{z}) + \lambda\|\mathbf{r}\|^2 + \ell(\mathsf{f}(\mathsf{c}(\mathbf{x}_o + \mathbf{r})), t), \tag{15}$$

where ϕ is defined by (6). In words, $\mathbf{z}$ represents an unconstrained perturbation, while $\mathbf{r}$ should be close to $\mathbf{z}$, smooth like $\mathbf{x}_o$, small, and such that the perturbed input $\mathbf{x}_o + \mathbf{r}$ satisfies the classification objective. Then, by letting $\mu \to \infty$, the first term becomes a hard constraint imposing a globally smooth solution at each iteration:

$$\min_{\mathbf{r},\mathbf{z}} \quad \lambda\|\mathbf{r}\|^2 + \ell(\mathsf{f}(\mathsf{c}(\mathbf{x}_o + \mathbf{r})), t) \tag{16}$$

$$\text{subject to} \quad \mathbf{r} = \hat{\mathsf{s}}_\alpha(\mathbf{z}), \tag{17}$$

where $\hat{\mathsf{s}}_\alpha$ is defined by (9). During optimization, every iterate of this perturbation $\mathbf{r}$ is smooth like $\mathbf{x}_o$.

4.2.3 Optimization

With this definition in place, we solve for $\mathbf{z}$ the following unconstrained problem over $\mathbb{R}^{n\times d}$:

$$\min_{\mathbf{z}} \quad \lambda \left\|\hat{\mathsf{s}}_\alpha(\mathbf{z})\right\|^2 + \ell(\mathsf{f}(\mathsf{c}(\mathbf{x}_o + \hat{\mathsf{s}}_\alpha(\mathbf{z}))), t). \tag{18}$$

Observe that this problem has the same form as (13), where $\mathbf{r}$ has been replaced by $\hat{\mathsf{s}}_\alpha(\mathbf{z})$. This implies that we can use the same optimization method as the C&W attack. The only difference is that the variable is $\mathbf{z}$, which we initialize by $\mathbf{z} = \mathbf{0}_{n\times d}$, and we apply function $\hat{\mathsf{s}}_\alpha$ at each iteration.

Gradients are easy to compute because our smoothing is a linear operator. We denote the loss on this new variable by $L(\mathbf{z}) := \ell(\mathsf{f}(\mathsf{c}(\mathbf{x}_o + \hat{\mathsf{s}}_\alpha(\mathbf{z}))), t)$. Its gradient is

$$\nabla_{\mathbf{z}} L(\mathbf{z}) = \mathbf{J}_{\hat{\mathsf{s}}_\alpha}(\mathbf{z})^\top \cdot \nabla_{\mathbf{x}} \ell(\mathsf{f}(\mathsf{c}(\mathbf{x}_o + \hat{\mathsf{s}}_\alpha(\mathbf{z}))), t), \tag{19}$$

where $\mathbf{J}_{\hat{\mathsf{s}}_\alpha}(\mathbf{z})$ is the $n \times n$ Jacobian matrix of the smoothing operator at $\mathbf{z}$. Since our smoothing operator is defined by (8) and (9) is linear, $\mathbf{J}_{\hat{\mathsf{s}}_\alpha}(\mathbf{z}) = \operatorname{diag}(\mathsf{s}_\alpha(\mathbf{1}_n))^{-1}\mathcal{L}_\alpha^{-1}$ is a matrix constant in $\mathbf{z}$, and multiplication by this matrix is equivalent to smoothing. The same holds for the distortion penalty $\left\|\hat{\mathsf{s}}_\alpha(\mathbf{z})\right\|^2$. This means that in the backward pass, the gradient of the objective (18) w.r.t. $\mathbf{z}$ is obtained from the gradient w.r.t. $\mathbf{r}$ (or $\mathbf{x}$) by smoothing, much like how $\mathbf{r}$ is obtained from $\mathbf{z}$ in the forward pass (17).

Matrix $\mathcal{L}_\alpha$ is fixed during optimization, depending only on input $\mathbf{x}_o$. For small images like in the MNIST dataset [31], it can be inverted: function $\hat{\mathsf{s}}_\alpha$ is really a matrix multiplication. For larger images, we use the *conjugate gradient* (CG) method [32] to solve the set of linear systems $\mathcal{L}_\alpha \mathbf{r} = \mathbf{z}$ for $\mathbf{r}$ given $\mathbf{z}$. Again, this is possible because matrix $\mathcal{L}_\alpha$ is positive-definite, and indeed, it is the most common solution in similar problems [26, 30, 33]. At each iteration, one computes a product of the form $\mathbf{v} \mapsto \mathcal{L}_\alpha \mathbf{v}$, which is efficient because $\mathcal{L}_\alpha$ is sparse. In the backward pass, one can either use CG on the gradient, or *auto-differentiate* (AD) through the forward CG iterations. We choose the latter because it is the simplest implementation-wise. The two options have the same complexity and should have the same run-time in theory. In practice, Tensorflow AD takes 0.43 s on average for 50 CG iterations on ImageNet and InceptionV3, while CG forward takes 0.33 s.

4.2.4 Discussion

The *clipping function* c that we use is just the identity over the interval $[0, 1]$, but outside this interval, its derivative is zero. Carlini & Wagner [8] therefore argue that the numerical solver of problem (13) suffers from getting stuck in flat spots: when a pixel of the perturbed input $\mathbf{x}_o + \mathbf{r}$ falls outside $[0, 1]$, it keeps having zero derivative after that and with no chance of returning to $[0, 1]$ even if this is beneficial. This limitation does not apply to our case thanks to the L_2 distortion penalty in (13) and to the updates in its neighborhood: such a value may return to $[0, 1]$ thanks to the smoothing operation.

5 Experiments

Our experiments focus on the *white box* setting, where the defender first exhibits a network, and then the attacker mounts an attack specific to this network, but we also investigate a *transferability* scenario. All attacks are *untargetted*, as defined by loss function (3).

5.1 Evaluation protocol

For the *perceptual evaluation of the quality* of the adversarial images, we follow the recommendation of [34]. This paper compares fifteen metrics (including SSIM, PSNR, and wPSNR) to the subjective perceptual evaluation of a panel of users. The conclusion is that *most apparent distortion* (MAD) [35] is the metric best reflecting user assessment. A low MAD score means better fidelity.

For *quantitative evaluation of the strength* of an attack, we use two global statistics and an operating characteristic curve. Given a test image set of N' images, we only consider its subset X of N images that are classified correctly without any attack. The accuracy of the classifier is N/N'. Let X_{suc} be the subset of X with $N_{\text{suc}} := |X_{\text{suc}}|$ where the attack succeeds and let $D(\mathbf{x}_o) := \|\mathbf{x}_a - \mathbf{x}_o\|$ be the distortion for image $\mathbf{x}_o \in X_{\text{suc}}$.

The global statistics are the *success probability* P_{suc} and *expected distortion* $\overline{D}$ as defined in Section 2, estimated by

$$P_{\text{suc}} = \frac{N_{\text{suc}}}{N}, \quad \overline{D} = \frac{1}{N_{\text{suc}}} \sum_{\mathbf{x}_o \in X_{\text{suc}}} D(\mathbf{x}_o), \tag{20}$$

with the exception that $\overline{D}$ here is the *conditional average distortion*, where conditioning is on success. Indeed, distortion makes no sense for a failure.

If $D_{\max} = \max_{\mathbf{x}_o \in X_{\text{suc}}} D(\mathbf{x}_o)$ is the maximum distortion, the *operating characteristic* function $\mathsf{P} : [0, D_{\max}] \to [0, 1]$ measures the probability of success as a function of a given upper bound D on distortion. For $D \in [0, D_{\max}]$,

$$\mathsf{P}(D) := \frac{1}{N} |\{\mathbf{x}_o \in X_{\text{suc}} : D(\mathbf{x}_o) \le D\}|. \tag{21}$$

This function increases from $\mathsf{P}(0) = 0$ to $\mathsf{P}(D_{\max}) = P_{\text{suc}}$.

It is difficult to define a fair comparison of *distortion targeting* attacks to *optimality targeting* attacks. For the first family, we run a given attack several times over the test set with different target distortion ϵ. The attack succeeds on image $\mathbf{x}_o \in X$ if it succeeds on any of the runs. For $\mathbf{x}_o \in X_{\text{suc}}$, the distortion $D(\mathbf{x}_o)$ is the minimum distortion over all runs. All statistics are then evaluated as above.

5.2 Datasets, networks, and attacks

5.2.1 MNIST [36]

We consider a simple convolutional network with three convolutional layers and one fully connected layer that we

denote as C4, giving accuracy 0.99. In detail, the first convolutional layer has 64 features, kernel of size 8 and stride 2; the second layer has 128 features, kernel of size 6 and stride 2; the third has also 128 features, but kernel of size 5 and stride 1.

5.2.2 ImageNet

We use the dataset of the NIPS 2017 adversarial competition [37], comprising 1000 images from ImageNet [38]. We use InceptionV3 [39] and ResNetV2-50 [40] networks, with accuracy 0.96 and 0.93 respectively.

5.2.3 Attacks

The following six attacks are benchmarked:

- L_∞ distortion: FGSM [41] and I-FGSM [10].
- L_2 distortion: An L_2 version of I-FGSM [42], denoted as PGD_2 (projected gradient descent).
- Optimality: The L_2 version of C&W [8].
- Smooth: Our smooth versions qPGD_2 of PGD_2 (Section 4.1) and sC&W of C&W (Section 4.2). Note that the smoothness constraint integration differs a lot between qPGD_2 and sC&W.

5.2.4 Parameters

On MNIST, we use $\epsilon = 0.3$ for FGSM; $\epsilon = 0.3, \alpha = 0.08$ for I-FGSM; $\epsilon = 5, \alpha = 3$ for PGD_2; confidence margin $m = 1$, learning rate $\eta = 0.1$, and initial constant

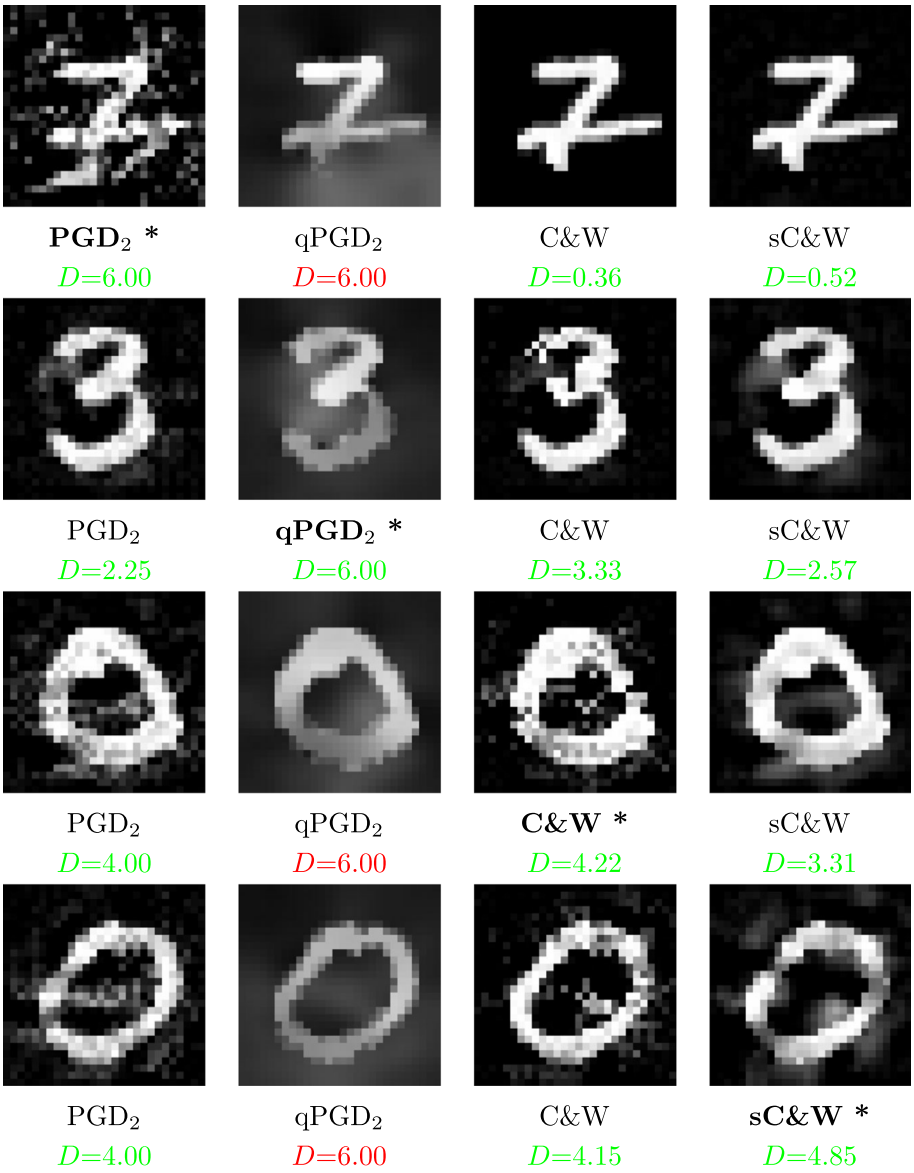


Fig. 2 For a given attack (denoted by an asterisk and bold typeface), the adversarial image with the strongest distortion D over MNIST. In green, the attack succeeds; in red, it fails

$c = 15$ (the inverse of λ in (4)) for C&W. For smoothing, we use Laplacian feature kernel, set $\alpha = 0.95$, and pre-compute $\mathcal{L}_\alpha^{-1}$. On ImageNet, we use $\epsilon = 0.1255$ for FGSM; $\epsilon = 0.1255, \alpha = 0.08$ for I-FGSM; $\epsilon = 5, \alpha = 3$ for PGD_2; $m = 0$, $\eta = 0.1$, and $c = 100$ for C&W. For smoothing, we use Laplacian feature kernel, set $\alpha = 0.997$, and use 50 iterations of CG. These settings are used in Section 5.5.

5.3 White box scenario

5.3.1 Qualitative results and perceptual evaluation

Figures 2 and 3 show MNIST and ImageNet examples, respectively, focusing on worst cases. Both sC&W and $qPGD_2$ produce smooth perturbations that look more natural. However, smoothing of $qPGD_2$ is more aggressive especially on MNIST, as these images contain flat black or white areas.

This is due to the "quick and dirty" integration of the smoothness constraint: On some images, the perturbation update $\hat{\mathbf{s}}_\alpha(\mathbf{g})$ is weakly correlated with gradient $\mathbf{g}$, which does not help in lowering the classification loss. Consequently, the perturbation becomes stronger in order to succeed. For the same reason, $qPGD_2$ completely fails on natural images like Fig. 3a and c. It consumes way more distortion than PGD_2, and although this perturbation is smoother, it becomes visible.

By contrast, the proper integration of the smoothness constraint in sC&W produces totally invisible perturbation. For images like Fig. 3a or c, sC&W consumes more distortion than C&W, but the perturbation remains less visible according to the MAD score. The reason is the "phantom" of the original that is revealed when the perturbation is isolated.

The superior perceptual quality of our smooth adversarial examples is also confirmed quantitatively: On ImageNet, 93% of the images produced by sC&W have lower MAD score than the ones by C&W. Figure 4 shows that when the MAD score of sC&W is greater than the one

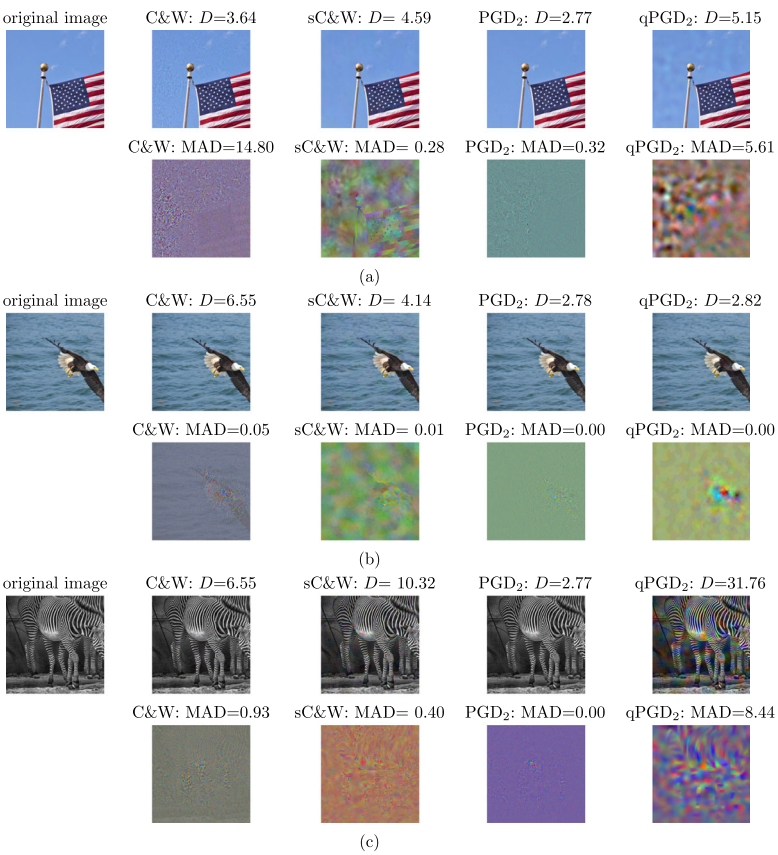


Fig. 3 Original image $\mathbf{x}_o$ (left), adversarial image $\mathbf{x}_a = \mathbf{x}_o + \mathbf{r}$ (above) and scaled perturbation $\mathbf{r}$ (below; distortion $D = \|\mathbf{r}\|$ and MAD scores) against InceptionV3 on ImageNet. Scaling maps each perturbation and each color channel independently to [0, 1]. The perturbation $\mathbf{r}$ is indeed smooth like $\mathbf{x}_o$ for sC&W. **a** Despite the higher distortion compared to C&W, the perturbation of sC&W is totally invisible, even when magnified (*cf* Fig. 1). **b** One of the failing examples of [6] that look unnatural to human vision. **c** One of the examples with the strongest distortion over ImageNet for sC&W: $\mathbf{x}_o$ is flat along stripes, reducing the dimensionality of the "smooth like $\mathbf{x}_o$" manifold

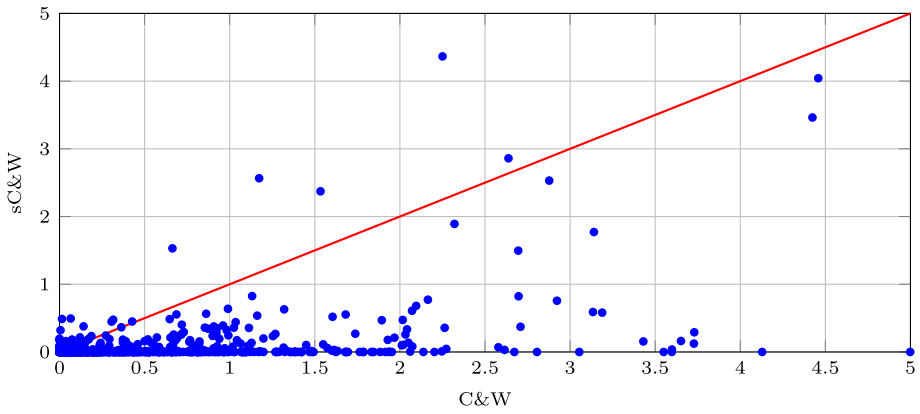


Fig. 4 MAD scores [35] of sC&W *vs.* C&W for all images of ImageNet. For 93% of the images below the diagonal, sC&W is less perceptible than C&W according to MAD score

of C&W, it usually happens for very small score values (below 0.1), meaning that both are almost equally imperceptible.

5.3.2 Quantitative results on success and distortion

The global statistics $P_{\text{suc}}, \overline{D}$ are shown in Table 1. Operating characteristics over MNIST and ImageNet are shown in Figs. 5 and 6 respectively.

We observe that our sC&W, with the proper integration via a latent variable (18), improves a lot the original C&W in terms of distortion, while keeping the probability of success roughly the same. This result, consistent in all experiments, is surprising. We would expect a price to be paid for a better invisibility as the smoothing is adding an extra constraint on the perturbation. This price can be rather high in the literature: In order to preserve the success rate, Table 1 of [7] reports an increase of distortion by a factor of 3 when integrating smoothness in the attack. An explanation may be that the smoothing operation of [7] is independent of the input image; while in our case, smoothing is guided by the input.

On the contrary, the "quick and dirty" integration (12) dramatically spoils qPGD$_2$ with big distortion especially on MNIST. This reveals the utmost importance of properly integrating the smoothness constraint. It cannot be just a post-processing filtering of the perturbation.

The price to pay for smoothing is the run-time: using Tensorflow and 50 CG iterations on ImageNet and InceptionV3, sC&W takes 205 s per image on average, while C&W takes 47 s. This is with our own implementation of CG without particular optimization effort. Runtime was not within our objectives.

We further observe that PGD$_2$ outperforms by a vast margin the C&W attack, which is supposed to be close to optimality. This may be due in part to how the Adam optimizer treats L_2 norm penalties as studied in [43]. This interesting finding is a result of our new evaluation protocol: C&W internally optimizes its parameter $c = 1/\lambda$ independently per image, while for PGD$_2$, we externally try a small set of target distortions D on the entire dataset. This is visible in Fig. 5, where the operating characteristic is piecewise constant. Our comparison is fair, given that C&W is more expensive.

As already observed in the literature, ResnetV2 is more robust to attacks than InceptionV3: The operating characteristic curves are shifted to the right and increase at a slower rate.

Table 1 Success probability P_{suc} and average L_2 distortion $\overline{D}$

	MNIST		ImageNet			
	C4		InceptionV3		ResNetV2	
	P_{suc}	$\overline{D}$	P_{suc}	$\overline{D}$	P_{suc}	$\overline{D}$
FGSM	0.89	5.02	0.97	5.92	0.92	8.20
I-FGSM	1.00	2.69	1.00	5.54	0.99	7.58
PGD$_2$	1.00	1.71	1.00	1.80	1.00	3.63
C&W	1.00	2.49	0.99	4.91	0.99	9.84
qPGD$_2$	0.97	3.36	0.96	2.10	0.93	4.80
sC&W	1.00	1.97	0.99	3.00	0.98	5.99

5.4 Adversarial training

The defender now uses adversarial training [41] to gain robustness against attacks. Yet, the white box scenario still holds: this network is public. The training set comprises images attacked with "step l.l" model [10][1]. The accuracy of C4 on MNIST (resp. InceptionV3 on ImageNet) is now 0.99 (resp. 0.94).

Table 2 shows interesting results. As expected, FGSM is defeated in all cases, while average distortion of all

[1] Model taken from https://github.com/tensorflow/models/tree/master/research/adv_imagenet_models

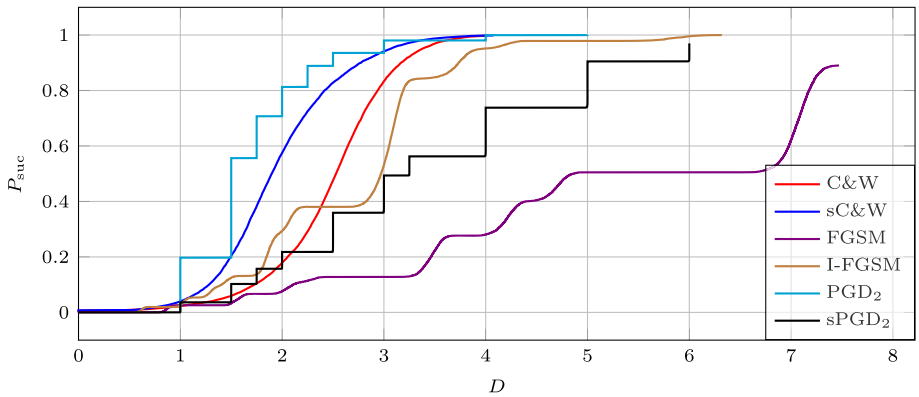


Fig. 5 Operating characteristics of the attacks over MNIST. Attacks PGD_2 and qPGD_2 are tested with target distortion $D \in [1, 6]$

attacks is increased in general. What is unexpected is that on MNIST, sC&W remains successful while the probability of C&W drops. On ImageNet however, it is the probability of the smooth versions qPGD_2 and sC&W that drops. I-FGSM is also defeated in this case, in the sense that average distortion increases too much.

5.5 Transferability

This section investigates the transferability of the attacks under the following scenario: the attacker has now a partial knowledge about the network. For instance, he/she knows that the defender chose a variant of InceptionV3, but this variant is not public so he/she attacks InceptionV3 instead. Also, this time, he/she is not allowed to test different distortion targets. The results are shown in Table 3.

The first variant uses a bilateral filter (with standard deviations 0.5 and 0.2 in the domain and range kernel respectively; *cf* Appendix 1) before feeding the network. This does not really prevent the attacks. PGD_2 remains a very powerful attack if the distortion is large enough. Smoothing makes the attack less effective, but the perturbations are less visible. The second variant uses the adversarially trained InceptionV3, which is, on the contrary, a very efficient counter-measure under this scenario.

Figure 7 shows the operating characteristics of C&W and sC&W corresponding to the bilateral filter results of Table 3. We see that within a distortion budget of 5, sC&W succeeds with 67% probability, whereas C&W with 50%. Yet, at larger distortion budgets, C&W keeps on forging more adversarial images whereas sC&W stops making progress. This is understandable: C&W creates strong artifacts clearly visible when the distortion is larger or equal to 5, as shown in Fig. 3. The upfront defense filters out some of these strong perturbations, but the rest remain successful. These images are adversarial by definition, yet not useful in practice because they are too much distorted.

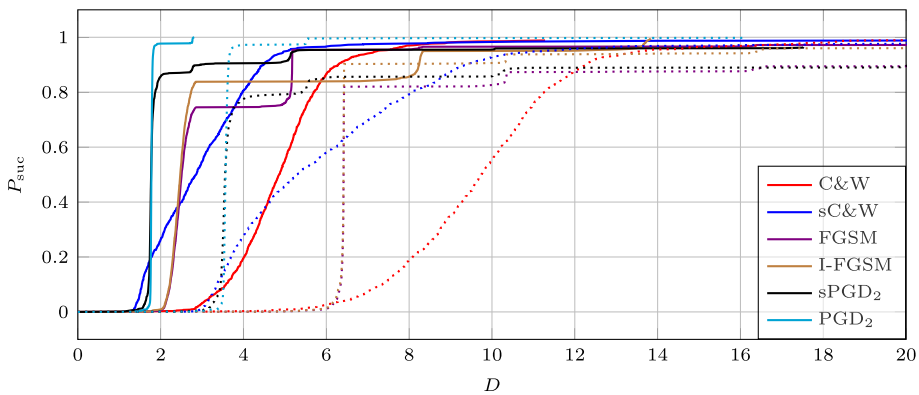


Fig. 6 Operating characteristics over ImageNet attacking InceptionV3 (solid lines) and ResNetV2-50 (dotted lines)

Table 2 Success probability and average L_2 distortion $\overline{D}$ when attacking networks adversarially trained against FGSM

	MNIST - C4		ImageNet - InceptionV3	
	P_{suc}	$\overline{D}$	P_{suc}	$\overline{D}$
FGSM	0.15	4.53	0.06	6.40
I-FGSM	1.00	3.48	0.97	29.94
PGD_2	1.00	2.52	1.00	3.89
C&W	0.93	3.03	0.95	6.43
qPGD_2	0.99	2.94	0.69	7.86
sC&W	0.99	2.39	0.75	6.22

6 Conclusion

Smoothing helps masking the adversarial perturbation by shaping it "like" the input image. However, this rule holds only when smoothness is properly integrated in the attack. Filtering the perturbation by post-processing is not a sound idea, even if it is done in accordance with the original image, even if it is done at each attack iteration. A sounder integration is to inject smoothness as a constraint inside the loss function.

It is impressive how sC&W improves upon C&W in terms of distortion and imperceptibility at the same time while maintaining the same success rate. To our knowledge and as far as a white box scenario is considered, this is the first time smoothness comes for free from this viewpoint. Yet, a price to be paid is the larger complexity.

Smoothing allows the attacker to delude more robust networks thanks to larger distortions while still being invisible. However, its impact on transferability is mitigated. The question raised in the introduction is still open: Fig. 1 shows that a human does not make the difference between the input image and its adversarial example even with magnification. This does not prove that an algorithm will not detect some statistical evidence.

Table 3 Success probability and average L_2 distortion $\overline{D}$ of attacks on variants of InceptionV3 under transferability

	Bilateral filter		Adv. training	
	P_{suc}	$\overline{D}$	P_{suc}	$\overline{D}$
FGSM	0.77	5.13	0.04	10.20
I-FGSM	0.82	5.12	0.02	10.10
PGD_2	1.00	5.14	0.12	10.26
C&W	0.82	4.75	0.02	10.21
qPGD_2	0.95	5.13	0.01	10.17
sC&W	0.68	2.91	0.01	4.63

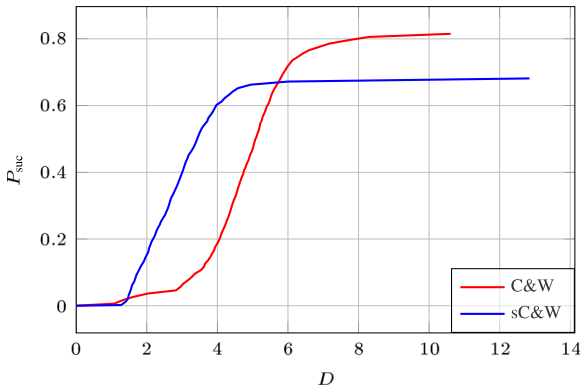


Fig. 7 Operating characteristics of C&W and sC&W on ImageNet with InceptionV3 under bilateral filter transferability, corresponding to Table 3

Appendix 1

Adversarial magnification

Given a single-channel image $\mathbf{x} : \Omega \to \mathbb{R}$ as input, its *adversarial magnification* $\mathsf{mag}(\mathbf{x}) : \Omega \to \mathbb{R}$ is defined as the following *local normalization* operation

$$\mathsf{mag}(\mathbf{x}) := \frac{\mathbf{x} - \mu_{\mathbf{x}}(\mathbf{x})}{\beta\sigma_{\mathbf{x}}(\mathbf{x}) + (1-\beta)\sigma_{\Omega}(\mathbf{x})}, \tag{22}$$

where $\mu_{\mathbf{x}}(\mathbf{x})$ and $\sigma_{\mathbf{x}}(\mathbf{x})$ are the local mean and standard deviation of $\mathbf{x}$ respectively, and $\sigma_{\Omega}(\mathbf{x}) \in \mathbb{R}^+$ is the global standard deviation of $\mathbf{x}$ over Ω. Parameter $\beta \in [0,1]$ determines how much local variation is magnified in $\mathbf{x}$.

In our implementation, $\mu_{\mathbf{x}}(\mathbf{x}) = \mathsf{b}(\mathbf{x})$, the *bilateral filtering* of $\mathbf{x}$ [44]. It applies a local kernel at each point $p \in \Omega$ that is the product of a domain and a range Gaussian kernel. The *domain kernel* measures the geometric proximity of every point $q \in \Omega$ to p as a function of $\|p-q\|$, and the *range kernel* measures the photometric similarity of every point $q \in \Omega$ to p as a function $|\mathbf{x}(p) - \mathbf{x}(q)|$. On the other hand, $\sigma_{\mathbf{x}}(\mathbf{x}) = \mathsf{b}_{\mathbf{x}}((\mathbf{x} - \mu_{\mathbf{x}}(\mathbf{x}))^2)^{-1/2}$. Here, $\mathsf{b}_{\mathbf{x}}$ is a *guided* version of the bilateral filter, where it is the reference image $\mathbf{x}$ rather than $(\mathbf{x} - \mu_{\mathbf{x}}(\mathbf{x}))^2$ that is used in the range kernel.

When $\mathbf{x} : \Omega \to \mathbb{R}^d$ is a d-channel image, we apply all the filters independently per channel, but photometric similarity is just one scalar per point as a function of the Euclidean distance $\|\mathbf{x}(p) - \mathbf{x}(q)\|$ measured over all d channels.

In Fig. 1, $\beta = 0.8$. The standard deviation of both the domain and range Gaussian kernels is 5.

Acknowledgements
Not applicable.

Authors' contributions
The authors declare equal contributions. The authors read and approved the final manuscript.

Funding
The Ph.D. thesis of Hanwei Zhang is funded by the Chinese Scholarship Council. Teddy Furon is funded by the chaire on artificial intelligence SAIDA.

Availability of data and materials
We use public datasets MNIST [36] and a subset of ImageNet [37], comprising 1000 images [38], and published networks InceptionV3 [39] and ResNetV2-50 [40].

Competing interests
The authors declare that they have no competing interests.

References

1. C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, R. Fergus, Intriguing properties of neural networks (2013). arXiv:1312.6199
2. A. Kurakin, I. Goodfellow, S. Bengio, Y. Dong, F. Liao, M. Liang, T. Pang, J. Zhu, X. Hu, C. Xie, J. Wang, Z. Zhang, Z. Ren, A. Yuille, S. Huang, Y. Zhao, Y. Zhao, Z. Han, J. Long, Y. Berdibekov, T. Akiba, S. Tokui, M. Abe, Adversarial attacks and defences competition (2018). arXiv:1804.00097
3. M. Sharif, L. Bauer, M. K. Reiter, On the suitability of l_p-norms for creating and preventing adversarial examples (2018). arXiv:1802.09653
4. S. Moosavi-Dezfooli, A. Fawzi, O. Fawzi, P. Frossard, in *CVPR*. Universal adversarial perturbations (IEEE, 2017), pp. 86–94
5. Z. Zhao, D. Dua, S. Singh, in *ICLR*. Generating natural adversarial examples, (2018). https://doi.org/10.18653/v1/d18-1316
6. W. Heng, S. Zhou, T. Jiang, Harmonic adversarial attack method (2018). arXiv:1807.10590
7. C. Guo, J. S. Frank, K. Q. Weinberger, Low frequency adversarial perturbation (2018). arXiv:1809.08758
8. N. Carlini, D. Wagner, in *IEEE Symp. on Security and Privacy*. Towards evaluating the robustness of neural networks, (2017). https://doi.org/10.1109/sp.2017.49
9. I. J. Goodfellow, J. Shlens, C. Szegedy, Explaining and harnessing adversarial examples (2014). arXiv:1412.6572
10. A. Kurakin, I. Goodfellow, S. Bengio, Adversarial examples in the physical world (2016). arXiv:1607.02533
11. S.-M. Moosavi-Dezfooli, A. Fawzi, P. Frossard, in *CVPR*. Deepfool: a simple and accurate method to fool deep neural networks, (2016). https://doi.org/10.1109/cvpr.2016.282
12. K. Crammer, Y. Singer, On the algorithmic implementation of multiclass kernel-based vector machines. J. Mach. Learn. Res. **2**(Dec), 265–292 (2001)
13. D. Kingma, J. Ba, Adam: a method for stochastic optimization (2015). arXiv:1412.6980
14. W. Zhou, X. Hou, Y. Chen, M. Tang, X. Huang, X. Gan, Y. Yang, in *ECCV*. Transferable adversarial perturbations (Springer International Publishing, Cham, 2018), pp. 471–486
15. E. Quiring, D. Arp, K. Rieck, in *2018 IEEE European Symposium on Security and Privacy (EuroS P)*. Forgotten siblings: unifying attacks on machine learning and digital watermarking, (2018), pp. 488–502. https://doi.org/10.1109/EuroSP.2018.00041
16. I. Cox, M. Miller, J. Bloom, J. Fridrich, T. Kalker, *Digital Watermarking, 2nd edn.* (Elsevier, 2008)
17. F. Luan, S. Paris, E. Shechtman, K. Bala, in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Deep Photo Style Transfer, (2017)
18. G. Puy, P. Pérez, A flexible convolutional solver with application to photorealistic style transfer (2018). arXiv:1806.05285
19. Y. Li, M.-Y. Liu, X. Li, M.-H. Yang, J. Kautz, A closed-form solution to photorealistic image stylization (2018). arXiv:1802.06474
20. J. Su, D. Vargas, K. Sakurai, One pixel attack for fooling deep neural networks. IEEE Trans. Evol. Comput. (2019). https://doi.org/10.1109/tevc.2019.2890858
21. D. Zhou, O. Bousquet, T. N. Lal, J. Weston, B. Schölkopf, in *NIPS*. Learning with local and global consistency (MIT Press, 2003), pp. 321–328
22. T. H. Kim, K. M. Lee, S. U. Lee, in *ECCV*. Generative image segmentation using random walks with restart, (2008). https://doi.org/10.1007/978-3-540-88690-7_20
23. A. Sandryhaila, J. M. Moura, Discrete signal processing on graphs. IEEE Trans. Sig. Process. **61**(7) (2013). https://doi.org/10.1109/tsp.2013.2238935
24. D. I. Shuman, S. K. Narang, P. Frossard, A. Ortega, P. Vandergheynst, The emerging field of signal processing on graphs: extending high-dimensional data analysis to networks and other irregular domains. IEEE Sig. Process. Mag. **30**(3) (2013). https://doi.org/10.1109/msp.2012.2235192
25. A. Iscen, Y. Avrithis, G. Tolias, T. Furon, O. Chum, in *CVPR*. Fast spectral ranking for similarity search, (2018). https://doi.org/10.1109/cvpr.2018.00796
26. L. Grady, Random walks for image segmentation. IEEE Trans. PAMI. **28**(11), 1768–1783 (2006)
27. P. Vernaza, M. Chandraker, in *CVPR*. Learning random-walk label propagation for weakly-supervised semantic segmentation, (2017). https://doi.org/10.1109/cvpr.2017.315
28. X. Zhu, Z. Ghahramani, J. Lafferty, in *ICML*. Semi-supervised learning using Gaussian fields and harmonic functions, (2003), pp. 912–919
29. D. Zhou, J. Weston, A. Gretton, O. Bousquet, B. Schölkopf, in *NIPS*. Ranking on data manifolds (MIT Press, 2003), pp. 169–176. http://papers.nips.cc/paper/2447-ranking-on-data-manifolds.pdf
30. A. Iscen, G. Tolias, Y. Avrithis, T. Furon, O. Chum, in *CVPR*. Efficient diffusion on region manifolds: recovering small objects with compact CNN representations, (2017). https://doi.org/10.1109/cvpr.2017.105
31. Y. Lecun, L. Bottou, Y. Bengio, P. Haffner, Gradient-based learning applied to document recognition. Proc. IEEE. **86**(11) (1998). https://doi.org/10.1109/5.726791
32. J. Nocedal, S. Wright, *Numerical Optimization*. (Springer, 2006). https://doi.org/10.1007/978-3-540-35447-5
33. S. Chandra, I. Kokkinos, in *ECCV*. Fast, exact and multi-scale inference for semantic image segmentation with deep Gaussian CRFs, (2016). https://doi.org/10.1007/978-3-319-46478-7_25
34. S. A. Fezza, Y. Bakhti, W. Hamidouche, O. Déforges, in *2019 Eleventh International Conference on Quality of Multimedia Experience (QoMEX)*. Perceptual evaluation of adversarial attacks for CNN-based image classification, (2019), pp. 1–6. https://doi.org/10.1109/QoMEX.2019.8743213
35. E. C. Larson, D. M. Chandler, Most apparent distortion: full-reference image quality assessment and the role of strategy. J. Electron. Imaging. **19**(1), 011006 (2010). https://doi.org/10.1117/1.3267105
36. Y. LeCun, C. Cortes, C. Burges, MNIST handwritten digit database. **2** (2010). AT&T Labs [Online]. Available: http://yann.lecun.com/exdb/mnist. Accessed 30 June 2020
37. A. Kurakin, I. Goodfellow, S. Bengio, Y. Dong, F. Liao, M. Liang, T. Pang, J. Zhu, X. Hu, C. Xie, et al., Adversarial attacks and defences competition (2018). arXiv:1804.00097
38. J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, L. Fei-Fei, in *CVPR*. Imagenet: a large-scale hierarchical image database (Ieee, 2009), pp. 248–255. https://doi.org/10.1109/cvpr.2009.5206848
39. C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, Z. Wojna, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. Rethinking the inception architecture for computer vision, (2016), pp. 2818–2826. https://doi.org/10.1109/cvpr.2016.308
40. K. He, X. Zhang, S. Ren, J. Sun, in *Computer Vision – ECCV 2016*, ed. by B. Leibe, J. Matas, N. Sebe, and M. Welling. Identity Mappings in Deep Residual Networks (Springer International Publishing, Cham, 2016), pp. 630–645. https://github.com/KaimingHe/resnet-1k-layers
41. I. J. Goodfellow, J. Shlens, C. Szegedy, in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*. Explaining and harnessing adversarial examples, (2015). http://arxiv.org/abs/1412.6572
42. N. Papernot, F. Faghri, N. Carlini, I. Goodfellow, R. Feinman, A. Kurakin, C. Xie, Y. Sharma, T. Brown, A. Roy, A. Matyasko, V. Behzadan, K. Hambardzumyan, Z. Zhang, Y.-L. Juang, Z. Li, R. Sheatsley, A. Garg, J. Uesato, W. Gierke, Y. Dong, D. Berthelot, P. Hendricks, J. Rauber, R. Long, Technical report on the cleverhans v2.1.0 adversarial examples library (2018). arXiv:1610.00768
43. I. Loshchilov, F. Hutter, Fixing weight decay regularization in adam. CoRR. **abs/1711.05101** (2017)
44. C. Tomasi, R. Manduchi, in *ICCV*. Bilateral filtering for gray and color images, (1998). https://doi.org/10.1109/iccv.1998.710815